Construction of a High-Level Internet Web Page to Monitor and Display Building Energy Performance

Victor Manuel Sanchez Jr.
United Negro College Foundation Special Program
North Carolina Agricultural and Technical State University
National Renewable Energy Laboratory
Golden, CO 80401

August 10, 2001

Prepared in partial fulfillment of the requirements of the Department of Energy UNCF Program under the direction of Michael Smith in the Thermal Test Facility at the National Renewable Energy Laboratory in Golden, Colorado

Participant:

Signature

Research Advisor:

Signature

Table of Contents

Abstract

Construction of a high level Web page with the purpose of monitoring a building and displaying the calculations and results collected from the selected building and displaying it on the World Wide Web. Victor Manuel Sanchez (North Carolina Agricultural and Technical State University, Greensboro, NC 27411) Michael Smith (National Renewable Energy Laboratory, Golden, CO 80401).

In the world of Internet Web page construction, many different languages cater to the needs of the programmer. Among them, Java (the object-oriented program language) from Sun Microsystems is generally capable of running on any computer without it being necessary to make any changes. Constructing the Web page using Java with HTML (HyperText Markup Language) enable us to have Web pages that can run and display data. The main function of the Web page is to display real-time data from one of NREL's Energy Efficient Buildings. The Building on which I decide to focus most my attention on is the Zion Visitor Center in Springdale, Utah. Simulated data was used for this project so that in the future real data can be pumped into the Web page. The data transfer occurs when the web page opens a file of numbers, which are separated by commas. These numbers are pumped into the file nearly every five minutes and then displayed on Applet frame every three seconds. The data consist of three categories: Outside Conditions, Wind factors, and Energy. These categories are displayed in the frame along with a photo of the Photovoltaic (PV) system from the Zion Visitor Center. Most of Zion's data will be collected from the PV system because the performance is mainly focusing on the energy aspect of the buildings. But due to the lack of actual real-time data the Web page will be published after the 2001 summer season. In the future the Web page will display actual data from a selected energy efficient building and real-time graphs will be accompanying the displayed real-time data.

Research Category for UNCF: Computer Science

School Author Attends: North Carolina Agricultural and Technical State University
DOE National Laboratory Attended: National Renewable Energy Laboratory (NREL)
Mentor's Name: Michael Smith
Task Leader: Paul Torcellini
Phone: (303) 384 - 7591
E-mail address: micheal_smith@nrel.gov

Presenter's Name: Victor Manuel Sanchez Jr.
Mailing Address: 4937 Beauregard St #102
City/State/ZIP: Alexandria, VA 22312
Phone: (703) 256 – 3550
E-mail Address: vs980977@ncat.edu

Introduction

Use of the World Wide Web to display data has been rising during the last eight years. Recently, there have been developments to display the data not just in numerical form but also in real-time charts, tables, and graphs. The purpose of this project is to display energy performance data from selected buildings numerically and visually on the World Wide Web, using real-time graphs. One might think that the construction of Web pages that display numerical data and graphics is a difficult task. But, in the world of Web-page construction, there are many methods and designing tools one can use to construct an effective Web site without too much stress. Today, construction tools cater to the programmer's immediate needs and long-term goals.

Java is currently one of the most popular programming languages used to construct Web sites that display numerical and visual data. Since the mid-1990s, Java has had a significant impact on the World Wide Web. Java was introduced in 1995 by Sun Microsystems and was instantly embraced by many Web developers (Cassady-Dorion, 2000). The main reason for Java's rapid acceptance is that programmers have always wanted software capable of running on any computer without needing modifications to the coding (Linden, 1999). Before Java, the mainstream coding languages for programmers were C++ and COBOL, which had to be modified depending on the system on which the code was compiled and run (Stanek, 1996). The Java programming language also has many different runtime models, which can be used to accomplish many different tasks.

Programmers use three main runtime application models in Java. These runtime applications are the applet application, the window application, and the console

application. The first and most common way of writing Java programs is the applet. The applet, which is the most secure of the runtime models, runs within a Web browser. The Java applets appear as a part of HTML (HyperText Markup Language) documents similar to a picture on an HTML document. The difference between the picture, which is a still graphic, and the applet is that the Java applet is a complete running program. One advantage of using the applet is that the programmer needs to produce only one copy of the class files on the HTML server. The class file is where the actual output is sent after the Java program is compiled and ran. Doing this eliminates the need to distribute and install software by tape or disk. The main disadvantage of using the Java applet runtime application is that the user must be using a Web browser that supports Java.

The second application model is window, which can be used like any window application. The window application is also referred to as graphic user interface (GUI) application. A GUI application can be opened, closed, minimized and maximized. For example, Microsoft Word, is a GUI application. An applet is just like the window application model but without the classic window frame (Figure 1). The two advantages of the window application are that it is capable of accessing local resources, and it can be integrated with non-Java applications. There are two disadvantages of this runtime application. The first one is that access to the local resources is unsecured, meaning that the code can be altered not only by the programmer but also by the users. The second disadvantage is that users of the program must have a local copy to use, meaning the actual code must be saved locally.

The third Java application model is the console application, which is executed from the command line, meaning that there are no windows opening and the only

communication with the program is through System methods. Examples of these methods are System.out.println(), which is used to print characters and integers on the screen, and System.in.readln(), which is used to input the lines into the program. Advantages of the console application model are that it, like the window model, has access to local resources and it can be integrated with non-Java applications. The disadvantages are also the same as those of the window application: access to the local resources is unsecured, and the user must have a local copy to use the program.

All of these runtime application models were used at one time during the construction of the Web page. During the early development stages, the console and window applications were used to make sure the source code was correct before the applet was launched. The console ensured that the file from which the information was being read from was up and running. The window application was used for all graphical components because the applet and frame are very similar. But for the final product of this project, the applet will be the only runtime application model used. The long-term goal of this project is to have a Web page that would collect data from different buildings and display real-time data and images. The short-term goal was to focus of the Zion Visitor Center Web page.

The Zion Visitor Center complex, located in Zion National Park in Springdale, Utah, is one of the National Park Service's most energy efficient complexes. This project was mainly focused on reporting the performance of the Visitor Center's photovoltaic system (Figure 1A), which provides some of the building's electric power. In addition to the PV system there are eight other major factors that combine to make the Zion Visitor Center very energy efficient. These factors include the trombe wall, lighting, glazing

design and selection, passive down-draft cooltower, energy-efficient landscape, natural ventilation, thermal mass flooring, and optimized overhangs (Figure 1B).

Due to the location, southern Utah, of Zion National Park power is not very reliable, and there is a need for a uninterrupted power supply (UPS). This Visitor Center PV system provides power to charge this UPS system. The PV panels, located on the south roof, provide about 20% of the electricity needed to power the Visitor Center. So the combination of utility power and the PV system, provides a very dependable power source for the Visitor Center. As far as heating the Zion Visitor Center does not depend on the PV systems, but mainly on the Trombe wall.

The Trombe wall heats the Visitor Center passively. Heat from the sun is trapped between a pane of glass and a black-coated masonry wall. The Trombe wall glass is made from glass that is single-glazed and has a high-transmittance. The masonry wall stores the heat from the sun then later in the day releases it the heat into the building. The Trombe wall is the lower two rows of the south wall, below the view windows (Figure 1C). The windows are also a big part of the Zion Visitor Center's efficiency, because they provide most of the lighting for the building. The windows' design allows the maximum amount of daylight to enter the building and at the same time minimizing the glare from direct sunlight during the hottest time of the year. This is accomplished using optimized overhangs, meaning the horizontal overhangs above the windows were engineered so that they completely shade the windows during the hot summer months when the sun is high in the sky. During the winter, when the sun is low in the sky, the sun is not blocked by the overhangs and is able to enter the buildings to be used to offset building heating loads. The overall result is that the overhangs help keep the building

cool in the summer and help heat the building in the winter. But the overhangs, even though they help a lot, are not the only factor that keeps the building cool during hot days. Also during the winter, the sunlight shines directly onto the Visitor Center's concrete floor. This method of having concrete floors is called thermal mass flooring. The idea came from the stone walls of Zion Canyon. The concrete is a thick material that absorbs and stores the sun's heat, resulting in the warm floor helping heat the building and maintain a comfortable temperature even after the sunsets.

Natural ventilation also helps cool the building during hotter days in southern Utah. Natural Ventilation cooling means outside air flows through the building to remove the warm inside air and replace it with cooler outside air. Sometimes natural ventilation is not enough to keep the building cool. When this happens, the passive down-draft cooltower (Figure 1D) helps bring the temperature down. Water is pumped over pads at the top of the cooltower evaporates to cool the air. The cool air is denser than the hot air in the building so the cool air naturally drops down through the tower and enters the building through large openings at the base of the tower.

Materials and Methods

The programming languages used to construct this Web-site project are Java and HTML. The Java editor JBuilder 4 made by Borland was used to compile and run Java applications. Along with the JBuilder 4 software package, a CD-ROM titled *Companion Tools* was used in the project. The *Companion Tools* CD-ROM contains a program called JB Chart that was used to make graphs for the Web site.

Construction of Basic Window Frame for Displaying Data from File

Every window application program created in Java must have four basic factors in order to work properly. First, the programmer must import the proper packages (Figure 2 – A) depending on the function of the program. Second, the file must have a main class (Figure 2 – B); this should be the same name as the filename of the program (e.g., filename.java). Third, the program must have a main method (Figure 2 – C). The main method is what creates the instances of the secondary class and initiates the program to run and go though future methods. Fourth, the program must have a separate class (Figure 2 – D) containing a constructor and a paint function. This second class is the class that holds all code for displaying data in the frame. This class is created by the main function in an instance and then made visible to be displayed in a frame. This class must indicate that the program is a frame by the "*extends*" tag that follows the name of the class (Figure 2 – D). Inside the class must be the constructor, which has the same name as the class in which it is enclosed. The purpose of the constructor (Figure 2 – E) is to set the size of the frame, set the color of fonts and drawings, set the background color, create all buttons (if needed), and set all variables created in the program to an initial value. In Java, unlike C++, where variables are created, the default value of the variable is zero. Also enclosed in the class is the paint function (Figure 2 – F). The purpose of the paint function is to display (or paint) data, rectangles, graphs, and the like onto the window frame for the user to view. All these factors are needed to run an error-free window frame.

Transformation of Java Window Application to an Applet

Though the window and applet applications are very similar in appearance, the code is very different. First, unlike the window application, a main method is not needed for the program to run. Second, there is a special package the programmer must import in order to use an applet application (Figure 3 – A). Next, like the frame, to show that the program is an applet, the tag "*extends"* must follow the declaration of the applet class (Figure 3 – B). Also unlike the window code, there is no need for a secondary class to be declared in the applet application, but there is a need to declare a constructor for the applet class (Figure 3 – C). Inside this constructor is where the programmer would throw exceptions, create buttons, and set variables to an initial value. The only factor that stays the same through window to applet transfer is the paint function (Figure 3 – D). The major change that happens when the programmer changes the window application into an applet is that the applet class is imported into an HTML file (Figure 3 – E). This allows the applet to be posted on an HTML document, and then it can be posted on the World Wide Web.

Results

Overall the project is successful in doing its main function, reading data from a file in real-time format. However, due to a bug in the project's applet code, the project was not transformed into a Java applet that reads from the file. As research went further, it was discovered that Java applets can not read from a text file like the console and GUI applications can. In order for an applet to successfully read in data from an outside source, the applet must be launched on the web. Then when the applet is on the web, the data must be read from a separate HTML file (Cassady-Dorion, 1997). Trying to

simulate this file did not work either for the Java applet. Inside of the application, the code is structured and commented to help the next programmer who takes the task of converting the GUI application into an applet.

Discussion and Conclusions

The construction of a Java web page can be a difficult task if one does not have superior knowledge of the programming language. Java has many loopholes and rules that many people are not knowledgeable of. The example in this case is the applet not being able to read numbers from a file. In the past, constructing the web page has been just the easy steps of converting one application to the other. But in this case, just a small process such as reading numbers from a file delayed a whole project. It's a computer programmer's fact that with a majority of programming languages, you learn as you go along. When first learning a different language, the professor/teacher installs the basic knowledge need to do simple task and how to avoid syntax errors. So while the programmer continuously codes, he/she picks up knowledge from their mistakes. Computer programming is something your have to love to do, because if one doesn't they will not be successful. Computer programmers learn from their mistakes and it takes time for one to fix the mistakes. But when the mistakes are discovered after three hours of frantic searching, the programmer never forgets the mistake and it will never happen again. In the case of this project, the programmer did not win his battle due to time.

Acknowledgements

I would like to thank the United States Department of Energy for giving me this opportunity to further my knowledge and gain experience working in the field of computer science. I would also like to thank the National Renewable Energy Laboratory

and the Center for Buildings and Thermal Systems for supporting and teaching me all summer. Through this summer I have learned so much about things I had never before been interested in, such as energy-efficient buildings. Not only did I sharpen my skills in Java programming, but I also gained an awareness of how much energy we are consuming in our everyday life and how much of a problem it is. Special thanks to Linda Lung for all the care and guidance they gave all the interns throughout the program. Also thanks goes to my mentor, Michael Smith, for helping me make the transition from East Coast living to living in Colorado. Last but not least, I would like to thank the United Negro College Foundation Special Programs and Laverne Means for giving me the opportunity to travel away from the East Coast for the first time and see how beautiful nature can be.

REFERENCES

1.) Van der Linden, P. (1996). Just Java. SunSoft Press A Prentice Hall Title, Mountain View, California

2.) Cassady-Dorion, L. (1997). Industrial Strength JAVA, For Serious Programmers, New Riders Publishing, Indianapolis, Indiana

3.) Stanek, W. R. (1996). HTML, CGI, SGML, VRML, JAVA Web Publishing Unleashed, Sams.net, Indianapolis, Indiana

Figure 1A – Zion Visitor Center's Photovoltaic System

Figure 1B – Features of Zion Visitor Center

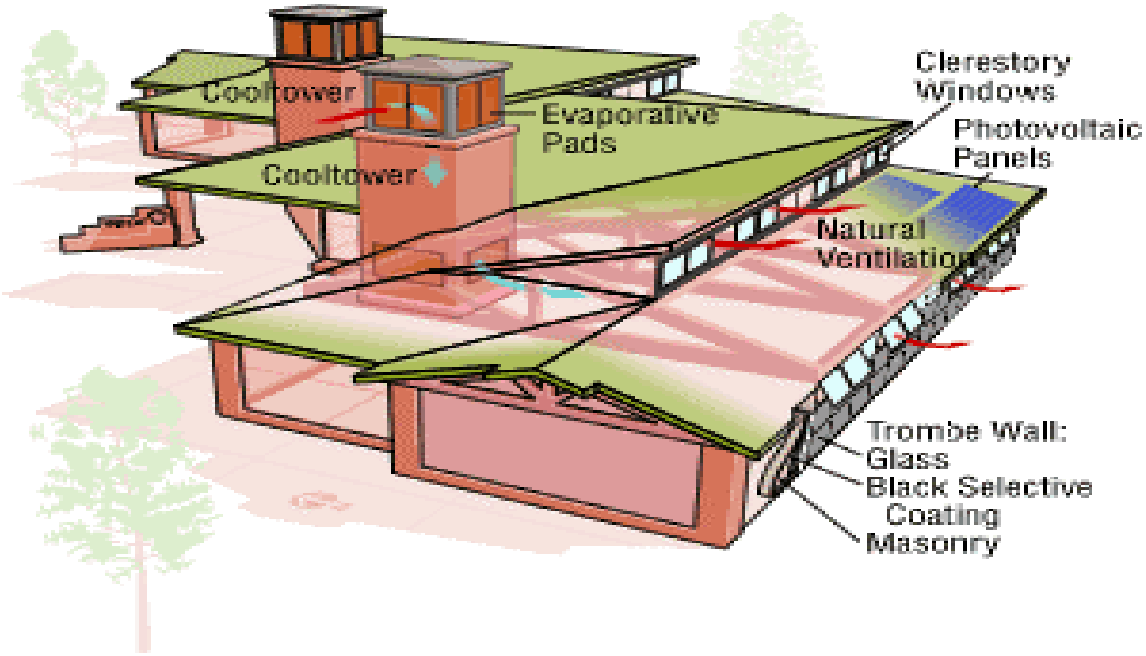

Figure 1C – Trombe Wall

Figure 1D – Passive Down-Draft Cooltower

Figure 2 – BasicWindowClass.java

```
import java.io.*;
import java.awt.*;
import java.util.*;
import java.lang.*;
import java.awt.event.*;

//*************************************************

class BasicWindowClass
{

 public static void main(String[] args)
 {
        BasicClass B = new BasicClass();   //created instance of the secondary class (BasicClass)
        B.setVisible(true);                //make sure that it is visible
 }//end of main method

}//end of class BasicWindowClass

//*************************************************

 class BasicClass extends Frame
 {
  public BasicClass()
  {
        //For frame creation
        super("Basic Window Frame");              //in blue box above frame
        final int width = 700, height = 650;
        setSize(width, height);                   //set size of frame
        setBackground(Color.black);               //set background color
        setForeground(Color.black);               //set color of drawings and font

        //this is where one would add Buttons and Textfield input

}//end of BasicClass constructor

 //*************************************************

 public void paint(Graphics g)
 {
        g.setFont(new Font("SansSerif",Font.BOLD,25));  //set font size and type to be painted

        //making rectangle
        g.setColor(Color.yellow);                 //sets color of font and drawings
        g.fillRect(355,350,320,200);              //creates solid yellow rectangle

        //painting string to be printed in rectangle
        g.setColor(Color.black);                  //the color of the font is now black
        g.drawString("Basic Window Frame", 400, 360);       //paints string

    }//end of paint method
```

Figure 3 – BasicApplet.java and BasicApplet.html

BasicApplet.java

```
import java.awt.*;
import java.lang.*;
import java.applet.*;
;


public class BasicApplet extends Applet
{
 public void init()
 {
        //where one would set font colors and size in applet

        setBackground(Color.black);         //setting background color
 }

//*************************************************************************

 public NRELApplet()
 {
        //if needed create buttons, throw exceptions, set variables

 }

//*************************************************************************

 public void paint(Graphics g)
 {
        g.setFont(new Font("SansSerif",Font.BOLD,30));
        g.drawString("Basic Applet Application",20,200);
 }//end of paint method

}//end of NRELApplet class
```

A

B

C

D

BasicApplet.html

```
<HTML>
<HEAD><TITLE>Basic Applet Application</HEAD></TITLE>
<APPLET CODE = "BasicApplet.class"  CODEBASE = "Applets"  Width = 300 HEIGHT = 300>
</APPLET>
</HTML>
```

E